In the Claims

1. (cancelled) A method for disabling clocks to at least one processor core of a plurality of processor cores comprising:

calculating an executing core limit based at least in part on a workload;

executing an n number of available threads; wherein n is an integer,

enabling an m number of processor cores, wherein m is an integer and is less than or equal to n, the number of available threads.

2. (cancelled) The method of claim 1 wherein disabling clocks to at least one processor during an idle time period as the processor core waits for a memory operation.

3. (cancelled)The method of claim 1 wherein disabling clocks to at least one processor core results in decreased power consumption.

4. (cancelled) The method of claim 1 wherein disabling clocks to at least one processor core allows for increasing the operating frequency of that processor core.

5. (withdrawn) A method for selecting a voltage and frequency operating point to at least one processor core of a plurality of processor cores comprising:

predicting an activity level of a plurality of threads running on all of the plurality of processor cores;

enabling a subset of the plurality of processor cores based at least in part on the activity level.

6. (withdrawn) The method of claim 1 wherein the activity level is an executing core

limit that is based at least in part on adhering to thermal power considerations.

7. (withdrawn) The method of claim 6 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; %E depicts the percentage executing time; and %M depicts the percentage memory reference time. and the formula is:

int (N x (%E/ (%E + %M)))

8. (withdrawn) A state diagram for a plurality of multi-core processors comprising:

A first state for a core without an assigned thread;

A second state for a queue to store cores with an assigned thread;

A third state for enabling the core to run a thread; and

A fourth state to disable a core.

9. (withdrawn) The state diagram of claim 8 wherein the queue is a first in first out (FIFO) queue.

10. (withdrawn) The state diagram of claim 8 wherein the core transitions from a second state to the third state if the number of enabled cores is less than an executing core limit.

11. (withdrawn) The state diagram of claim 10 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; %E depicts the percentage executing time; and %M depicts the percentage memory reference time. and the formula is :

int (N x (%E/ (%E + %M)))

12. (withdrawn) The state diagram of claim 8 wherein the core transitions from a third state to the fourth state if the core is idle as it waits for completion of a memory operation.

13. (withdrawn) A method for a state diagram for a plurality of multi-core processors comprising:

assigning a first state to a core without an assigned thread;

assigning a second state for a queue to store cores with an assigned thread;

comparing the number of enabled cores to an executing core limit, assigning a third state for enabling the core to run a thread if the number of enabled cores is less than the executing core limit; and

assigning a fourth state to disable a core.

14. (withdrawn) The method of claim 13 wherein the queue is a first in first out (FIFO) queue.

15. (withdrawn) The method of claim 13 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; %E depicts the percentage executing time; and %M depicts the percentage memory reference time. and the formula is :

int (N x (%E/ (%E + %M)))

16. (withdrawn) The state diagram of claim 13 wherein the core transitions from a third state to the fourth state if the core is idle as it waits for completion of a memory operation.

17. (withdrawn) A system of multi-core processors comprising:

at least one multi-core processor coupled to a cache memory, and coupled to at least two clockwise directional busses to receive requests and responses; and

a core rationing logic to manage the number of enabled cores to be less than or equal to an executing core limit.

18. (withdrawn) The system of claim 17 wherein the executing core limit is based at least in part on a formula, wherein N depicts the number of threads that have context; %E depicts the percentage executing time; and %M depicts the percentage memory reference time. and the formula is :

int (N x (%E/ (%E + %M))).

19. (withdrawn) The system of claim 17 further comprises a system interface that contains:

a plurality of memory controllers for memory DIMMs;

a router logic to handle the interconnection links to other processor dies or I/O subsystems; and

the core rationing logic.

20. (withdrawn) The system of claim 17 further comprises at least two counter-clockwise directional busses to receive requests and responses.

21. (withdrawn) The system of claim 17 wherein the cache memory is a level three (L3) memory with a plurality of independent memory banks.

22. (new) A method for disabling clocks to at least one processor core of a plurality of

processor cores comprising:

asserting a signal to one processor core based at least in part on executing an operation that is dependent on a previous memory operation, the signal to disable the clocks to the processor core;

blocking an instruction issue operation in response to the signal; and

transitioning the core to a cache coherent low power state of operation.

23. (new) The method of claim 22 further comprising defining an operating point based at least in part on an executing core limit and a voltage and frequency pair, and adjusting the operating point based on analysis of a number of cores status with respect to a waiting or rationing queue.

24. (new) A method for disabling clocks to at least one processor core of a plurality of processor cores comprising:

asserting a signal to one processor core based at least in part on executing an operation that is dependent on a memory operation, the signal to disable the clocks to the processor core;

blocking an instruction issue operation in response to the signal;

transitioning the core to a cache coherent low power state of operation;

assigning an identifier for the disabled processor core; and

de-asserting the signal for the disabled processor core upon completion of the memory operation if the number of executing cores is less than or equal to a predetermined executing core limit.

25. (new) The method of claim 24 further comprising defining an operating point based at least in part on an executing core limit and a voltage and frequency pair, and adjusting the operating point based on analysis of a number of cores status with respect to a waiting or

rationing queue.

26. (new) A core rationing logic to disable clocks to at least one processor core of a plurality of processor cores comprising:

a workload circuit to calculate a executing core limit;

a transmitter circuit to assert a signal to one processor core based at least in part on executing an operation that is dependent on a memory operation, the signal to disable the clocks to the processor core; and

a comparison circuit to de-assert the signal for the disabled processor core upon completion of the memory operation if the number of executing cores is less than or equal to the executing core limit.

27. (new) The logic of claim 26 further comprising the workload circuit to define an operating point based at least in part on an executing core limit and a voltage and frequency pair, and adjusting the operating point based on analysis of a number of cores status with respect to a waiting or rationing queue.

28. (new) A core rationing logic to disable clocks to at least one processor core of a plurality of processor cores comprising:

a workload circuit to calculate a executing core limit;

a transmitter circuit to assert a signal to one processor core based at least in part on executing an operation that is dependent on a memory operation, the signal to disable the clocks to the processor core;

a comparison circuit to de-assert the signal for the disabled processor core upon completion of

the memory operation if the number of executing cores is less than or equal to the executing core limit.

an assignment circuit to assign an identifier for the disabled processor core; and

the transmitter circuit to de-asserting the signal for the disabled processor core upon completion of the memory operation if the number of executing cores is less than or equal to a predetermined executing core limit.

29. (new) The logic of claim 28 further comprising the workload circuit to define an operating point based at least in part on an executing core limit and a voltage and frequency pair, and adjusting the operating point based on analysis of a number of cores status with respect to a waiting or rationing queue.